IN THE SPECIFICATION:

Please delete paragraph [0008] and replace it with the following new paragraph:

[0008] Figure 1 shows a simplified form of a data switch, Figure 2 shows an ~~egress~~ ingress multiplexer;

Please delete paragraph [0016] and replace it with the following new paragraph:

[0016] Figure 2 shows the architecture of an ingress multiplexer. An ingress multiplexer receives a set of data streams from the data sources via a set of low-bandwidth input ports. Each data stream is a sequence of equal size cells (that is, an equal number of bits of data). A set of N low-bandwidth ports 21 ~~each fills~~ is provided such that each low-bandwidth port fills one of the N input queues 22. An ingress control unit 23 extracts the destination address from each of the cells in the input queues and transfers them into a set of M virtual output queues 24. There is one virtual output queue for each low-bandwidth output port in the switch. The ingress multiplexer also contains an interconnect link control unit 25 which implements this function by scheduling cells from the virtual output queues 24 across the high-bandwidth link 26 to the central interconnect 1 according to an M-entry egress table 28.

Please delete paragraph [0025] and replace it with the following new paragraph:

[0025] The egress port table defines how the bandwidth of a high-bandwidth port to the central interconnect 1 is allocated across the virtual output queues. There is no issue with self-consistence as all possible entries are self-consistent[[.]] so that the bandwidth allocation for a virtual output queue v with weight w_v is given by:

$$p_f = \frac{w_v}{\sum_{n=0}^{(N-1)} w_n}$$

Please delete paragraph [0031] and replace it with the following new paragraph:

[0031] Despite the apparent complexity of this function, it may be implemented exclusively with an adder, multiplexers and small lookup tables, thus meeting

the requirement for [[sped]] speed and efficiency. Features of this weighting function are that, for $s_l = 1.0$, $s_s = 0.0$ and $s_u = 0.0$, bandwidth is allocated locally purely on the basis of queue length, ith a non-linear function, so that the switch always attempts to avoid queues overflowing. When $s_l = 0.0$, $s_s = 1.0$ and $s_u = 0.0$, bandwidth is allocated purely on the basis of pseudo-static allocations [[a]] as described above. Finally, when [[.]]$s_l = 0.0$, $s_s = 1.0$ and $s_u = 0.5$, bandwidth is allocated on the basis of pseudo-static allocation but a data source is allowed to "push" some data harder, when the demand arises, by setting the urgency bit in the appropriate cell headers.